CWO 48
SUBSYSTEM INTERFACE VERIFIER
SOFTWARE REQUIREMENTS DOCUMENT
FOR THE
DEEP SPACE NETWORK DATA SYSTEMS
SOFTWARE DEVELOPMENT PROGRAM

DSN-SIV-0-OPA-0-SRD/1.1

September 13, 1994

Prepared by:

TELOS SYSTEMS GROUP

Prepared for:

JET PROPULSION LABORATORY
PASADENA, CALIFORNIA

Contract No. 958442

THIS PAGE INTENTIONALLY LEFT BLANK

Deep Space Network

Subsystem Interface Verifier

DSN Operations Interfacing Software (DOI)

Software Requirements Document

Prepared by:

Laura Krall Date

Released by:

Peggy Panda Date
TDA/DSN Document Control

Approved by:

Laura Krall Date
Software Cognizant Development Engineer

Chaw-Kwei Hung Date
Task Manager

Joseph A. Wackley Date
Office 430 Implementation Engineering Manager

TDA/DSN No. SRD-DOI-5249-TP Rev. A

For Software Ver. OP-A

Issue Date September 13, 1994

NASA Ident. No. 23835

Jet Propulsion Laboratory

California Institute of Technology

Review Page

Reviewed by:

Required Reviewers

None.

Additional Reviewers

None.

CHANGE INCORPORATION LOG					
DOCUMENT STATUS			LIST PAGES AFFECTED	ISSUE DATE	SECTION
OP	Rev	Other			
A	A	Draft	ALL (? total pages)		398

DISTRIBUTION

Cureton-Snead, I.	525-3680	
DSN Program Library	503-102	(2)
DTF-21	125-B17	
Hung, C.	525-3682	
Krall, L.	525-3684	
Osoro, J.	507-120	
Salazar, A.	303-403	
Soderstrom, T.	525-3684	
SPMC	525-3681	
Stipanuk, J.	525-3681	
Vault	111-B25	(2)
Wackley, J.	303-403	

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 1

INTRODUCTION

1.1 Identification

Program Set Name:	Subsystem Interface Verifier (SIV)
Identification Number:	DOI-5249-TP
Version Number:	OP-A
Build Number:	2
Release Number:	1.1
Pseudonym:	SIV
Project:	SPC Upgrade Task
Subsystem:	Automated Test Software
Assembly:	Subsystem Interface Verifier (SIV)

1.2 Overview

The Deep Space Network (DSN) provides worldwide continuous realtime coverage of spacecraft activities to numerous users. DSN's requirement to provide its customers with almost 100% of reliable data is a continuous challenge. On it path to the end user, the raw data is processed through multiple subsystems whose interfaces are governed by written specifications. Uniform adherence to these interface agreements is crucial as even minimal data loss can threaten DSN's data delivery requirement.

On-going software changes to support new requirements compete for resources and must be reliably verified before they are used for project support. On the average, a DSN assembly has six external software interfaces. Complete verification of there interfaces requires support by the projects and DSN development personnel. Testing of the interfaces is prone to error, frequently resulting in the need to re-build and re-test the software. This is an inefficient and costly use of valuable project and DSN resources.

The Subsystem Interface Verifier (SIV) will provide a means to simulate and test subsystem interfaces as defined in DSN interface agreements (820-16 and 820-13). Such interfaces will initially be limited to those defined for data transmissions on the DSCC SPC LAN using the DSN standard 890-131 and DSN Network Data Flow Standard (DFL-1-1) protocols.

The SIV will be used to create bit-level interface data, essentially prototyping an interfacing subsystem without the expense of creating special simulation code for that subsystem. It will be used to receive data from a subsystem, dump it in a readable format, and validate the contents. This will allow interface testing to be performed and errors corrected as soon as interface agreements reach Level 3. Benefits of the SIV include early discovery of interface problems, decreased cost and complexity of

interface testing, and improved schedule performance.

The SIV will be developed and delivered in successive builds and initiated with a prototyping effort. Customer demonstrations will be held to demonstrate functionality of the SIV prior to delivery of each build. Each build will focus on one or more subsystems currently under development which would best benefit from use of the SIV in the corresponding time frame.

1.3 Document Scope

This document contains the software requirements for the SIV Program Set through Build 2. These requirements are written in response to the Statement of Work and feedback from users throughout prior SIV builds. In some instances, reference to requirements anticipated for future SIV builds will be made accompanied by an appropriate footnote.

1.4 Method

Jet Propulsion Laboratory Software Management Standards (D-4000) were followed in developing the software requirements documentation. Yourdon techniques were used in defining the software functional components of the SIV.

1.5 Notation

Data flow and context diagrams are presented using Ward-Mellor structured development methods.

1.5.1 Diagrammatic Notation

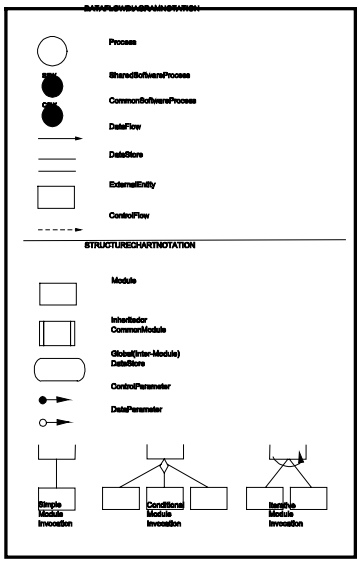

Figure 1-1 Diagrammatic Notation Example

1 illustrates the diagrammatic notations used in this document as explained below.

External entities are identified by rectangular boxes. Such entities might include external subsystems or assemblies of those subsystems.

Processes or functions are represented by bubbles or circles. Solid line bubbles

represent processes or transformations while dashed line bubbles represent control or decision processes.

Data flows are represented by solid lines, control flows with dashed lines, and direction of flow with arrowheads. Data flows provide definition of the continuity of information and/or control parameters passed between processes.

Data stores are represented by parallel lines with the name of the data store shown between the lines. Only primary data stores are represented in the data flow diagrams.

1.5.2 Special Notations

Requirements will be noted by using "will" as the flag word. Also requirements are set forth in the "processing" paragraphs of Section 4 (Program Set Functional Requirements).

1.6 Controlling Documents

D-4000 JPL Software Management Standards; Version 3; December, 1988

D-4005 JPL Software Management Standards; Software Requirements Analysis Phase; Version 3; December, 1988

1.7 Applicable Documents

890-131 DSCC General Data Flow Standards; Rev C; April 15, 1992

890-132 DSCC Monitor and Control Data Interchange Standards; Rev C; June 5, 1990

890-19, DFL-1-1 DSN Network Data Flow Standard; TBD

820-16 DSN System Requirements - Mark IVA Detailed Subsystem Interface Design; Modules for subsystems whose interfaces conform to SIV requirements for support.

Structured Development for Real-Time Systems; Ward, Mellor

1.8 Priorities

This paragraph defines the criteria by which requirements are assigned to three priority levels. These priorities are used below in Section 1.9, Functional Requirements, and in Appendix A, Traceability.

Priority 1:	CRITICAL	Any requirement which is necessary for SIV operation, and/or greatly facilitates or expands SIV usability.
Priority 2:	IMPORTANT	Any requirement necessary to support test of the targeted interface specifications.
Priority 3:	DESIRABLE	All remaining requirements.

1.9 Functional Requirements

The following functional capabilities describe the minimal requirements for SIV implementation. This list serves as a 'mini-FRD' as no such document exists for the SIV. The functional requirements below are organized by the following areas:

a. Network Communications
b. Data Block Generation
c. Data Block Reception & Validation
d. Monitor and Control
e. Miscellaneous

Each requirement is given an implementation priority value.

1.9.1 Network Communications

Number	Functional Requirement	Priority
NC1	Ethernet connection to subsystem under test via the SPC LAN or equivalent LAN will be supported.	1
NC2	Physical connection to subsystem under test via TBD serial interface will be supported.	3
NC3	Standard DSCC local facility protocol (890-131) will be supported.	1
NC4	Standard DSN Network Level protocol (DFL-1-1) will be supported.	1

1.9.2 Data Block Generation

Number	Functional Requirement	Priority
DG1	Create data blocks according to the formats and values described in the DSN Detailed Subsystem Interface Agreement documents (820-13 and 820-16).	1
DG2	Provide a means to generate interface data blocks such that data in each field assumes values that span the valid range specified in the interface agreement.	1
DG3	Provide a means to generate data that is outside the valid range specified in the interface agreement for each field.	1
DG4	Provide a means to generate coupled data for fields in interface blocks that are interrelated.	3
DG5	Provide a means to generate data for fields in interface blocks based on an algorithm.	3
DG6	Automate the generation of interface data blocks.	1
DG7	Send interface data blocks to the subsystem under test.	1
DG8	Rates of sending interface data blocks will be within the rate ranges specified in the interface agreement.	1
DG9	Generate interface data blocks for multiple interfaces of one subsystem concurrently.	2
DG10	Generate interface data blocks from raw (binary or ASCII) data files.	1

1.9.3 Data Block Reception and Validation

Number	Functional Requirement	Priority
RV1	Receive interface data blocks of 890-131 or DFL-1-1 format as sent from the subsystem under test.	1
RV2	Validate received data blocks for conformity to the formats and values described in the interface agreements.	1
RV3	Receive interface data blocks for multiple interfaces of one subsystem concurrently.	3
RV4	Print validation results in readable ASCII report format.	1
RV5	Print report of testing performed.	2
RV6	Provide optional data block logging to disk.	2

1.9.4 Monitor and Control

Number	Functional Requirement	Priority
MC1	Control the subsystem under test: i.e., monitor/control by DMC will not be required for testing.	1
MC2	Allow user to control testing or select displays via directives.	1
MC3	Allow user to control testing or select displays via a graphical interface.	3
MC4	Create a readable dump of blocks input or output to/from the subsystem under test.	1
MC5	Provide means to control testing: i.e., start, stop, configure, change level of reporting.	2
MC6	Capture responses from subsystem under test (event notices).	1
MC7	Log responses from subsystem under test.	1
MC8	Provide means to modify interface data dynamically during testing.	1
MC9	Provide status information regarding data flow including, minimally, block size, rate, and direction.	1
MC10	Provide a facility to repeat test steps under automatic control.	2
MC11	Allow user to select interface level to test with (eg, CM'd, Level 3, engineering, etc.).	2
MC12	Provide for dynamic visibility of data block contents.	1

1.9.5 Other Miscellaneous

Number	Functional Requirement	Priority
OT1	Will be ported to the following host platforms: MODCOMP 9735 under REAL/IX; SUN Workstation under Solaris 2.x; others TBD.	2
OT2	The SIV will generate Monitor Data definition files used by MSW from Monitor Data interfaces defined by RIDs	3
OT3	Utilize existing software to extent feasible.	1
OT4	Allow knowledgeable user to generate Interface Definition inputs using a standard text editor on a PC or Unix Workstation.	1
OT5	Allow knowledgeable user to generate Interface Definition inputs using a GUI on a platform TBD.	3

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 2

PROGRAM SET DESCRIPTION

The function of the Subsystem Interface Verifier (SIV) Program Set is to provide test services for verification of interface agreements between subsystems resident on the DSCC SPC LAN.

The SIV tool will capitalize on existing simulation and multiuse software to produce a program set appropriate for verification of interface agreements which conform to the 890-131 or DFL-1-1 protocols over the DSCC SPC LAN. The SIV will provide for data stream generation, reception, validation, and test automation. With SIV, developers can simplify and speed up their inter-subsystem testing.

1 is the context diagram of the SIV program set.

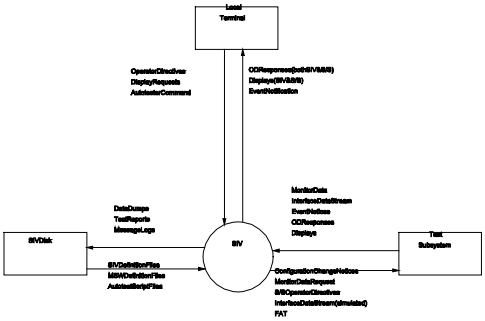

Figure 2-1 Context Diagram

2.1 Operational Concept

The SIV is being developed as a test tool to aid in verification of a subsystem's interface software. It is anticipated that both Cognizant Development Engineers (CDEs) and Cognizant Operations Engineers (COEs) will utilize the SIV at various stages of interface testing. Initially, CDEs will use the SIV to perform bit-level test of their interface data both for inbound and outbound flows. Through simple SIV directives, a CDE can initiate and control test data flows, monitor the data content, and generate test reports. As testing progresses, subsystem-specific tests can be further built through SIV interactive sessions and automated test scripts. A set of test scripts can be achieved such that a CDE can easily repeat tests as needed for regression testing.

2.2 Operational Environment

SIV is a test and simulation tool and will not operate in the DSN Operational Environment. The SIV will be made available to CDEs and COEs for testing subsystem software prior to their transfer to operations.

The SIV is projected to operate both in software test laboratories and at the Development and Test Facility (DTF-21) located at JPL in Building 125. SIV software will be made available for download into a provided machine at a particular subsystem's test laboratory provided the platform is supported by SIV. SIV software will be developed for portability to multiple platforms. A complete SIV system (workstation and SIV software) will reside at the DTF-21 facility for scheduled use. 1 illustrates the anticipated configuration of the SIV for both of these environments.

The SIV communicates with the interfacing subsystem via the SPC LAN. Monitor

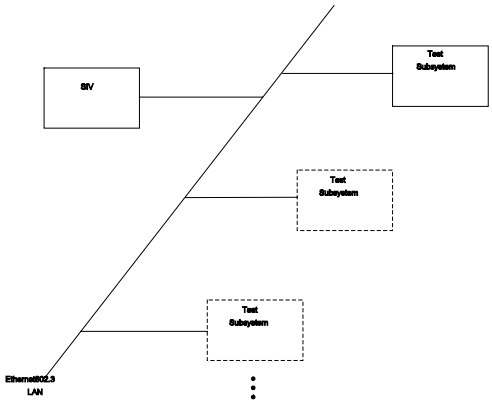

Figure 2-2 SIV Configuration at DTF-21 or Test Lab

and Control of the subsystem under test will be provided through SIV emulation of the DMC. A terminal interface to the SIV will be provided for the user interface.

2.3 Operational Modes

SIV will operate under the following modes: initialization, standby, configured, and test in progress. Each reflects the state of communications with the interfacing subsystem under test.

The subsections to follow detail each the SIV operation within each mode. 3 shows

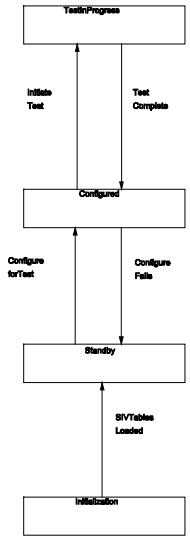

Figure 2-3 SIV Operating Modes

the SIV modes and valid transitions between each mode.

2.3.1 Initialization Mode

When SIV is initially activated, it will enter an initialization state whereby data definition tables are loaded and tasks are initialized for operation. Once all tasks within the SIV are ready to operate, the SIV will enter the 'Standby Mode' whereby SIV is ready to be configured for communication with the subsystem to be tested.

2.3.2 Standby Mode

While in the Standby Mode, the SIV can not yet communicate with the subsystem to be tested and awaits configuration by the user. This is the point at which the subsystem to be tested must be identified in order to obtain the necessary subsystem-specific information (e.g., LAN address, interface definitions, test scripts, etc.). Once received, the SIV will attempt to establish a dedicated communication path with the interfacing subsystem to insure that no interference from the DMC or other LAN subsystems can occur. (Note: The subsystem under test may require a reboot if previously FAT locked by a subsystem other than SIV -- e.g., DMC.) Provided the interfacing subsystem responds properly to test startup communications, the SIV will enter the Configured Mode; otherwise, the Standby mode will remain in effect.

2.3.3 Configured Mode

The SIV Configured Mode is entered once communication with the interfacing subsystem has been successfully achieved. At this point, the SIV is ready for test activation through operator directive (or automated test script).

This mode is also re-entered at the completion of a test (see below). The SIV is still configured for test with the interfacing subsystem allowing for quick re-run of the prior test.
Also within this mode, the SIV can be configured for test with another subsystem. Similar to the Standby mode, the appropriate configuration directive is entered and failure returns the SIV to the Standby Mode.

2.3.4 Test in Progress Mode

Once an interface test has been initiated through operator directive, the SIV enters and operates in the Test in Progress Mode. Within this mode, interface data is exchanged and validated. The specific interface to be tested and corresponding test activities are dependent upon directives entered manually or through the automated test script capability. Once the test is terminated or runs to completion, the SIV returns to the Configured Mode ready to start another test or to reconfigure for

another subsystem.

SECTION 3

INTERFACE REQUIREMENTS

The SIV will provide the capability to communicate with test subsystems via the SPC LAN (or test lab LAN) using standard DSN Intersubsystem Communication protocols. An user interface will be provided to monitor and control the SIV and interfacing subsystem.

3.1 Interfaces to External Subsystems

SIV will interface to many external subsystems. No formal interface agreement is required between the SIV and these subsystems as the SIV functions to test 'operational interfaces'.

The SIV will provide the capability to generate the following types of ISBs for transfer over the SPC LAN to the interfacing subsystem:

a. Output: Functional Address Translation (FAT)
Trigger: Configuration of the SIV for a specified subsystem.
Dependency: Proper identification of subsystem information.
Frequency: Asynchronous
Purpose: Used to establish a dedicated communication path between the interfacing subsystem and the SIV with lock to exclude other LAN residents.

b. Output: Configuration Change Notice (CCN)
Trigger: Configuration of the SIV for a specified subsystem.
Dependency: Successful transmission of FAT.
Frequency: Asynchronous
Purpose: Used to direct state transitions of the interfacing subsystem in order to bring it to the configuration necessary for test data flows to proceed.

c. Output: Monitor Segment Request (MDR)
Trigger: Configuration of the SIV for a specified subsystem.
Dependency: Successful transmission of subsystem assign CCN.
Frequency: Periodic poll
Purpose: Request the Program Identification Monitor Data segment from the interfacing subsystem in order to perform watch-dog monitoring of the subsystem's software.

d. Output: Operator Directive (OD)
Trigger: Detection of subsystem directive.
Dependency: Subsystem continues to respond to watch-dog monitoring.
Frequency: Asynchronous
Purpose: Used to send subsystem directives as if entered at the DMC in support of subsystem interface tests. Specific OD syntax and functionality need not be known by the SIV. OD is simply encapsulated and transferred to the interfacing subsystem.

e. Output: Display Graphics Control (DGC)
Trigger: Detection of subsystem display request.
Dependency: Subsystem continues to respond to watch-dog monitoring.
Frequency: Asynchronous
Purpose: To request or terminate a specified display as if entered at the DMC.

f. Output: Interface Defined 890-131 Formatted Blocks
Trigger: Request to generate specified interface stream.
Dependency: Subsystem continues to respond to watch-dog monitoring.
Frequency: As defined in SIV Interface Definition for block.
Purpose: To allow subsystem to test reception of 890-131 blocks as if from real interface as defined in interface agreement.

g. Output: Interface Defined DFL-1-1 Formatted Blocks
Trigger: Request to generate specified interface stream.
Dependency: Subsystem continues to respond to watch-dog monitoring.
Frequency: As defined in SIV Interface Definition for block.
Purpose: To allow subsystem to test reception of DFL-1-1 blocks as if from real interface as defined in interface agreement.

The SIV will be capable of accepting the following types of ISBs from an interfacing subsystem over the SPC LAN:

a. Input: Operator Directive Response (DR)
Trigger: As a result of an OD transfer to the interfacing subsystem.
Dependency: Time-out has not occurred since OD was transferred.
Frequency: Asynchronous
Purpose: Indicates rejection or completion to an operator directive.

b. Input: Event Notification (EN)
Trigger: None
Dependency: None
Frequency: Asynchronous
Purpose: Provides alarm or advisory messages during the course of subsystem operation for inclusion in the SIV's event log.

c. Input: Display Graphic (DG)

Trigger: After DGC transfer to the interfacing subsystem.
Dependency: TBD
Frequency: Asynchronous
Purpose: Provides the detailed display as specified by the Display Graphics Control.

d. Input: Display Graphic Rejection (DGR)
Trigger: After DGC transfer to the interfacing subsystem.
Dependency: Time-out has not occurred since DGC was transferred.
Frequency: Asynchronous
Purpose: Provides reason for the inability to generate the specified display.

e. Input: Interface Agreements Defined 890-131 Formatted Blocks
Trigger: Subsystem generates blocks.
Dependency: None
Frequency: As generated by subsystem.
Purpose: To validate 890-131 interface data generated by subsystem under test.

f. Input: Interface Defined DFL-1-1 Formatted Blocks
Trigger: Subsystem generates blocks.
Dependency: None
Frequency: As generated by subsystem
Purpose: To validate DFL-1-1 interface data generated by subsystem under test.

3.2 SIV Interface to User

The SIV will provide the capability to process the following types of inputs from the SIV user interface:

a. Input: Operator Directives
Trigger: User entry.
Dependency: SIV initialization completed.
Frequency: Asynchronous
Purpose: Provide user control of the SIV:
- Assembly and test configuration
- Modification of content and/or control of generated data
- Modification of validation criteria for a given input data stream
- Initiation and control of automated tests
- Generation and view control of SIV displays

- Transfer of subsystem directives/display requests
- Debug operations for visibility into software activity

b. Input: Display Request
   Trigger: User entry.
   Dependency: SIV initialization completed.
   Frequency: Asynchronous
   Purpose: Request a display (see outputs for list).

c. Input: Autocontrol Test Script
   Trigger: User directive.
   Dependency: Test Script file exists.
   Frequency: Asynchronous
   Purpose: Initiate automated test sequence.

The SIV will provide the capability to output the following to the SIV user interface:

a. Output: Operator Directive Response
   Trigger: User directive entry.
   Dependency: Output device is available.
   Frequency: Asynchronous
   Purpose: Indicates rejection or completion to an operator directive.

b. Output: Display
   Trigger: User request for display.
   Dependency: Output device is available.
   Frequency: Asynchronous
   Purpose: Provides the requested display:
   - Status and configuration summary displays
   - Data flow statistics per protocol
   - Data content for both inbound and outbound flows
   - Disk directory information and corresponding table/file contents
   - Status/results of interface test progression (both manual and automated tests)
   - Event notifications and other message logs
   - Visibility into values generated.

c. Output: Display Generation Rejection (DGR)
   Trigger: Erroneous display request from user.
   Dependency: Output device is available.
   Frequency: Asynchronous
   Purpose: Indicates rejection to a display generation control.

d. Output: Event Notification
Trigger: SIV processing event.
Dependency: Output device is available.
Frequency: Asynchronous
Purpose: Provides alarm or advisory message to the terminal.
- Status and configuration changes
- Test activities and controls performed
- Data generation/validation parameter manipulations
- Hardware or software error
- ENs forwarded from interfacing subsystem

3.3 Interfaces to Hardware Devices

3.3.1 Interface To Disk

The SIV will provide the capability to process the following types of inputs from a disk storage media:

a. Input: SIV Tables
Trigger: SIV Initialization
Dependency: Media is accessible
Frequency: Asynchronous
Purpose: Initialize SIV information for displays, directives, shared memory, etc.
Modify: Automatically by software as needed or in response to operator inputs

b. Input: Subsystem Information Tables
Trigger: Configuration directive
Dependency: Media is accessible
Frequency: Asynchronous
Purpose: Configure SIV for communication to and interfaces with subsystem under test.
Modify: Off-line and/or on-line through user interface

c. Input: SIV Interface Definition (RID) Files
Trigger: Stream activation or Data Reception
Dependency: SIV is configured for test and media is accessible
Frequency: Asynchronous
Purpose: Load specified interface definition data.
Modify: Off-line by text editor and/or on-line through user interface

d. Input: Autocontrol Test Scripts
Trigger: Test activation
Dependency: SIV is configured for test and media is accessible

Frequency: Asynchronous
Purpose: Load test directives for automated processing.
Modify: Off-line by text editor

e. Input: Raw Data File
Trigger: Stream activation
Dependency: The RID file being processed includes a reference to the raw data file
Frequency: Asynchronous
Purpose: Generate raw data blocks, simulate CMC down load of support data, and other generic uses involving predetermined static data.
Modify: Off-line (source of change not important)

The SIV will provide the capability to process the following types of outputs to the disk storage media:

a. Output: Raw Data Block (dump)
Trigger: Data Reception
Dependency: Logging is enabled and media is accessible.
Frequency: Asynchronous
Purpose: Store for later ingest or for analysis.

b. Output: HEX/ASCII Dump
Trigger: Data Reception
Dependency: Logging is enabled and media is accessible.
Frequency: Asynchronous
Purpose: Store for later analysis or display.

c. Output: ASCII Validation Report
Trigger: Data Reception
Dependency: Validation is enabled and media is accessible.
Frequency: Asynchronous
Purpose: Store for later analysis, display, or print.

b. Output: ASCII Log Files
Trigger: Corresponding data is generated.
Dependency: Logging is enabled and media is accessible.
Frequency: Asynchronous
Purpose: EN Message Log, Autocontrol Test Log, Terminal Log, stored for later analysis or display.

3.3.2 Interface to Other Devices

Review of industry standard device communications (e.g., RS232, IEEE 488) will be made for potential support of hardware interface testing later SIV build(s).

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 4

PROGRAM SET FUNCTIONAL REQUIREMENTS

This section details the software functional requirements of the Subsystem Interface Verifier (SIV) Program Set. Since SIV will maximize use of existing software, some design knowledge is inherent in the following processes. In order to emphasize SIV functionality, MSW processes will not be detailed but will be shown where needed to add clarity to SIV data flows.

1 is the Data Flow Diagram for the SIV Program Set.

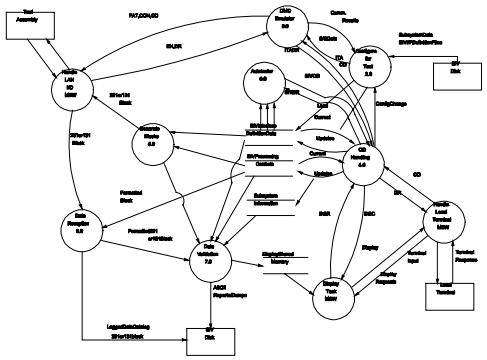

Figure 4-1 Level 1 Data Flow Diagram

Subsequent sections are organized by SIV processes as follows:

4.1 System Initialization (Process 1.0)

The SYSTEM INITIALIZATION process is not shown in Figure 4-1 as it unnecessarily would complicate the data flow diagram.

The SYSTEM INITIALIZATION process provides for the activation of SIV tasks. It consists of a MSW service which loads task executables from disk into memory as defined by initialization data files. Once loaded, SIV tasks are activated and individually initialize themselves for operation. A MSW synchronization process awaits acknowledgment from each SIV task before allowing subsequent operation. Once released from this hold, the SIV program set enters the Standby Mode ready for configuration.

2 is the Data Flow Diagram for this process.

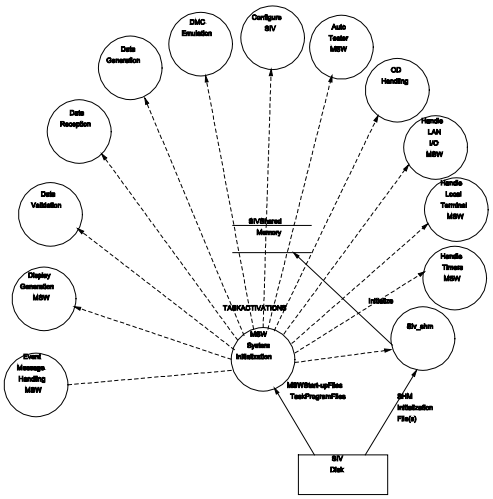

Figure 4-2 System Initialization

Inputs:

SOURCE	DATA
Disk	Task Executables
Disk	MSW Initialization Files
Disk	SIV Shared Memory Files

Processing:

The SYSTEM INITIALIZATION process will:

a. MSW performs corresponding system initialization services.
b. SIV shared memory process establishes data stores and initializes their content based on tables loaded from disk.
c. SIV processes reach Standby Mode in preparation for the CONFIGURATION process.

Outputs:

DESTINATION	DATA
Local Terminal	Initialization Messages
SIV Shared Memory	Default values

4.2 Configure for Test (Process 2.0)

The CONFIGURE FOR TEST process prepares the SIV and interfacing test assembly (ITA) for test activities.

3 is the Data Flow Diagram for this process.

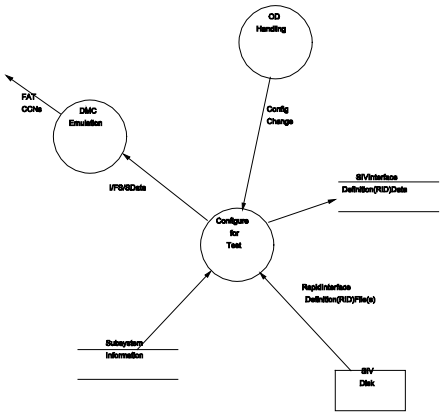

Figure 4-3 Configure for Test

Inputs:

SOURCE	DATA
Subsystem Info. Data Store	ITA Unique Information
Disk	RID File Data for ITA
OD HANDLING	Test Configuration Change Request

Processing:

The CONFIGURE FOR TEST process will:

a. Test Configuration change requests are accepted and processed.
b. Subsystem information for the interfacing test assembly (ITA) is read.
c. SIV configuration for prior ITA, if any, is removed.
d. The corresponding SIV Interface Definition (RID) files are loaded into data storage.
e. Subsystem information for the ITA is sent to the DMC EMULATION process.
f. Provide error checking of RID files.
g. Provide logging of errors and warnings to a disk file.

Outputs:

DESTINATION	DATA
RID Data Store	RID File Data for ITA
DMC EMULATION	ITA Unique Information
RID Log File	Error and Warning Messages

4.3 DMC Emulation (Process 3.0)

DMC EMULATION handles interactions with the interfacing test assembly (ITA) which would typically be performed by the DMC subsystem.

FAT and CCN blocks are sent to the ITA to establish a dedicated communication path over the LAN.

MD is requested and processed from the ITA to perform watch-dog checks on that assembly in order for SIV to detect its current operating state.

Subsystem ODs are sent to the ITA and corresponding response DRs processed.

Subsystem Display Generator Controls (DGC) are sent to the ITA and corresponding ITA Displays or Display Reject blocks are processed. Displays can be directed to SIV display screens.

ITA Event Notices are received and sent to either the MSW EN Service or MSW Autotester for corresponding output to the appropriate message log.

4 is the Data Flow Diagram for this process.

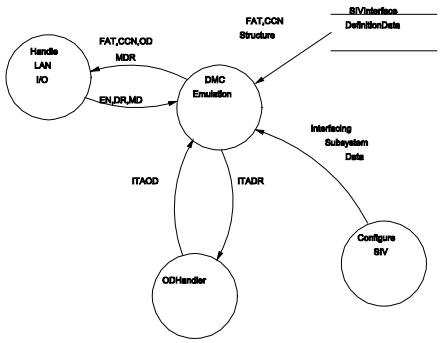

Figure 4-4 DMC Emulation

Inputs:

SOURCE	DATA
CONFIGURE FOR TEST	ITA unique data
RID Data Store	ITA FAT and CCN Structures
RID Data Store	MDR structure
OD HANDLING	ITA OD
HANDLE LAN I/O	ITA EN

Processing:

The DMC EMULATION process will:

a. FAT and CCN structures are extracted from the RID Data Store.
b. FAT and CCN blocks are formatted using ITA unique data received from the CONFIGURE FOR TEST process and sent to the MSW HANDLE LAN I/O process for output to the LAN.
c. A MDR for the ITA program ID is built, encapsulated for 890-131, and sent to the MSW HANDLE LAN I/O process for output to the LAN.
d. ODs destined for the ITA are received, encapsulated for 890-131, and forwarded to the MSW HANDLE LAN I/O process.
e. ENs, DRs, and MD are received, processed, and forwarded, when necessary, to the appropriate process.

Outputs:

DESTINATION	DATA
HANDLE LAN I/O	FAT
HANDLE LAN I/O	CCN
HANDLE LAN I/O	MDR
HANDLE LAN I/O	ITA OD
OD HANDLING	ITA DR
AUTOTESTER	ITA DR
AUTOTESTER	ITA EN
MSW EN	ITA EN

4.4 Operator Directive Handling (Process 4.0)

ODs received from the local terminal allow for control of SIV functions, change of SIV test configuration, and change to certain RID data.

5 is the Data Flow Diagram for this process.

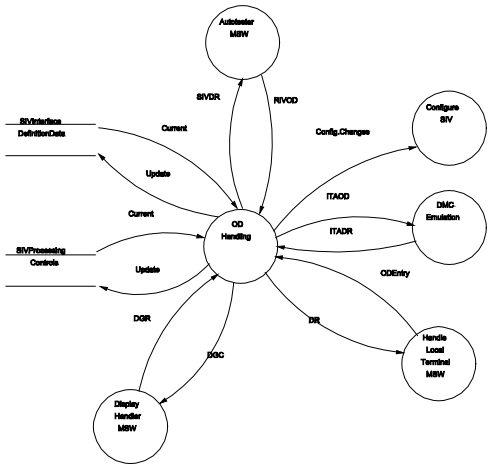

Figure 4-5 OD Handling

Inputs:

SOURCE	DATA
MSW HANDLE LOCAL TERMINAL	OD
AUTOTESTER	OD
MSW DISPLAY	DGR
DMC EMULATION	ITA DR
SIV Processing Controls Store	Current Control Params
RID Data Store	Current RID Data

Processing:

The OD HANDLING process will:

a. Accept directives from either the MSW HANDLE LOCAL TERMINAL process or the AUTOTESTER process.
b. Process directives to change the SIV configuration.
c. Forward information to the CONFIGURE FOR TEST process for completion of change request.
d. Process directives to view or change RID Data Store information.
e. Process directives to view or change parameters within the SIV Processing Controls Store.
f. Accept and forward display requests to the MSW DISPLAY HANDLER process.
g. Send directive responses to the appropriate MSW HANDLE LOCAL TERMINAL or AUTOTESTER process.
h. Process directives to change link assignment modes.
i. Process directive for setting communication mode.

Outputs:

DESTINATION	DATA
MSW HANDLE LOCAL TERMINAL	DR
AUTOTESTER	DR
DMC EMULATION	ITA OD
CONFIGURE FOR TEST	Config Change Data
RID Data Store	Updated RID Data
SIV Processing Control Store	Updated Control Params

4.5 Data Block Generation (Process 5.0)

The DATA BLOCK GENERATION process dynamically builds data blocks based on RID data, formats the block in the defined 890-131 or DFL-1-1 structure, and sends the blocks over the LAN to the ITA.

6 is the Data Flow Diagram for this process.

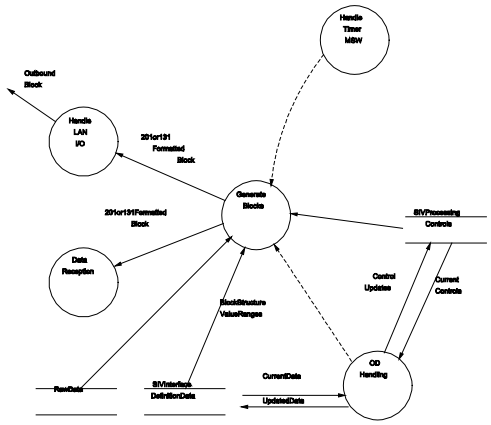

Figure 4-6 Data Block Generation

Inputs:

SOURCE	DATA
RID Data Store	Block Structure
RID Data Store	Current Data Values
RID DATA Store	Generation Controls
SIV Processing Controls Store	Generation Controls
MSW HANDLE TIMER	Expired Timer

Processing:

The DATA BLOCK GENERATION process will:

a. Extract RID data and build corresponding outbound data block.
b. Format the data block for 890-131 or DFL-1-1 transmission as defined by the RID data.
c. Send the formatted data block to the MSW HANDLE LAN I/O process for output to the LAN.
d. Send the formatted data block to the DATA RECEPTION process for visibility operations when enabled.
e. Respond to block generation controls (i.e., block generation rate, min/max/random value determination, etc.) within RID data unless specifically overridden by SIV Processing Controls.
f. Respond to SIV Processing Controls governing block output including suspend, resume, step, and log controls.
g. Provide block segmenting of a raw data file (ASCII or binary) stored on disk. The capability can be used to simulate CMC download of support data.
h. Schedule transmission of a data block at a given specified time.
i. Fill last block of raw data with either a word fill character or a portion of the data from the first block.
j. Support transmission variable transmission rates to at least a tenth of a second resolution.

Outputs:

DESTINATION	DATA
MSW HANDLE LAN I/O	Formatted Block
DATA RECEPTION	Formatted Block

4.6 Data Block Reception (Process 6.0)

The DATA BLOCK RECEPTION process accepts and decodes data blocks received from the LAN in the 890-131 or DFL-1-1 format. The raw data is also logged to disk.

7 is the Data Flow Diagram for this process.

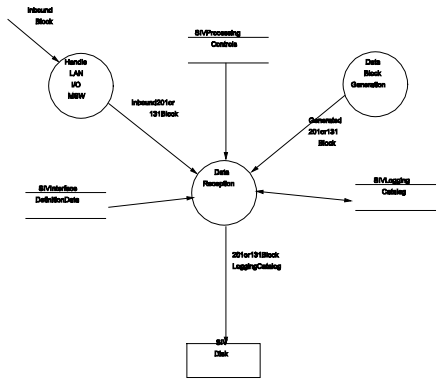

Figure 4-7 SIV Data Reception

Inputs:

SOURCE	DATA
MSW HANDLE LAN I/O	890-131 Formatted Block
MSW HANDLE LAN I/O	DFL-1-1 Formatted Block
SIV Processing Controls Store	Reception Controls

Processing:

The DATA BLOCK RECEPTION process will:

a. Accept inbound 890-131 or DFL-1-1 data blocks and perform initial decode.
b. Log raw data to disk if enabled and optionally strip off header data.
c. Respond to SIV Processing Controls affecting data reception (i.e., raw logging).
d. Identify RID file for a given block and store for validation access.
e. Maintain catalog of logged data.
f. Support variable data reception rates to at least a tenth of a second between blocks.
g. Timestamp reception of each block.

Outputs:

DESTINATION	DATA
Disk	Raw Data Block
SIV Logging Catalog	Extracted Data Block
	Store Information to identify logged data

4.7 Data Validation (Process 7.0)

The DATA VALIDATION process accepts and validates incoming 890-131 and DFL-1-1 blocks.

8 is the Data Flow Diagram for this process.

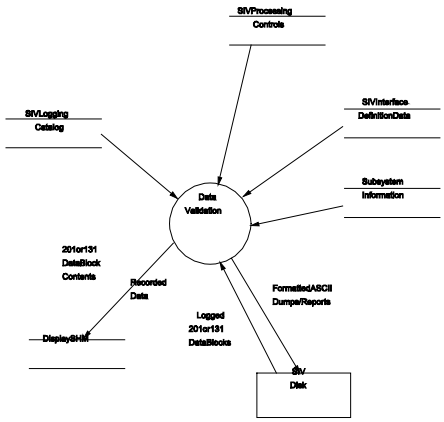

Figure 4-8 Data Validation

Inputs:

SOURCE	DATA
Disk	Logged 890-131 Block
Disk	Logged DFL-1-1 Block
SIV Data Store	Value Limits
SIV Data Store	Validation Controls
SIV Processing Controls Store	Validation Controls
Subsystem Information Store	ITA Unique Data
Subsystem Information Store	Validation Controls

Processing:

The DATA VALIDATION process will:

a. Accept incoming data blocks for data validation.
b. Identify corresponding RID data which defines the block's structure.
c. Limit check those fields within the data block having range definition.
d. Generate an ASCII dump and/or validation report when enabled.
e. Store the pertinent block data into shared memory for use by the MSW DISPLAY task in generating requested displays.
f. Respond to block validation controls contained within the RID data unless specifically overridden by SIV Processing Controls.
g. Respond to SIV Processing Controls which govern data validation (eg., validation on/off, dump on/off).
h. Respond to interface test assembly (ITA) validation controls.
i. Optionally validate DFL-1-1 header content against RID and GCF routing table.

Outputs:

DESTINATION	DATA
Display Data Store	Data Block
Display Data Store	Validation Stats
Disk	ASCII Reports
Disk	ASCII Dumps

4.8 Automated Testing (Process 8.0)

The AUTOMATED TESTING process is a modified version of the MSW AUTOTESTER. In addition to the functionality already provided, new directives to identify ITA directives, the ability to wait for new conditions (e.g., data block reception), and the interaction with the DMC EMULATION process will be added for

the version used in SIV.

The existing MSW AUTOTESTER functionality will not be shown to any detail. Rather, attention will be given to SIV specific flows.

9 is the Data Flow Diagram for this process.

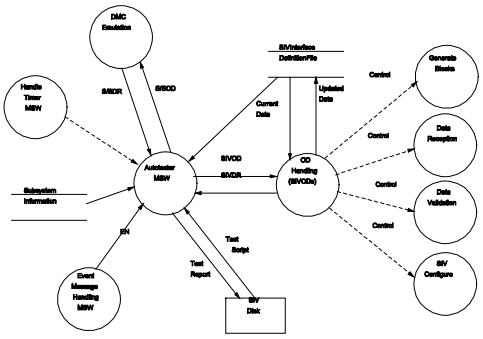

Figure 4-9 SIV Automated Testing

Inputs:

SOURCE	DATA
Disk	Test Script
RID Data Store	Generation Params
RID Data Store	Validation Params
RID Data Store	Reception Params
RID Data Store	Generation Controls
RID Data Store	Validation Controls
RID Data Store	Reception Controls
Subsystem Information Store	ITA Unique Data
OD HANDLING	SIV DR
DMC EMULATION	ITA DR
MSW HANDLE TIMER	Expired Timer

Processing:

The AUTOMATED TESTING process will:

a. Locate, read in, and process test script.
b. Respond to commands and data in test script.
c. Read and/or modify RID Data Store parameters and controls as per commands in test script.
d. Access required ITA unique data from Subsystem Information store as needed to perform commands.
e. Send SIV directives to OD HANDLING process to service.
f. Send ITA directives to DMC EMULATION process for forwarding and response monitoring.
g. Accept ITA and SIV DRs and process accordingly.
h. Respond to changes in Generation, Reception, and Validation controls.
i. Operate independently from other SIV processes.

Outputs:

DESTINATION	DATA
DMC EMULATION	ITA OD
OD HANDLING	SIV OD

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 5

PERFORMANCE REQUIREMENTS

5.1 SIV Configuration

The SIV will configure for an interface test within 1 second of the receipt of the configuration change request.

The SIV will support 50? interface agreement definitions per test subsystem.

5.2 Data Block Generation

The SIV will minimally support data block generation of 1 block per tenth of a second for blocks up to the maximum size of a LAN packet.

The SIV will respond to data block generation processing controls within 1 second of the receipt of the request.

The SIV will support 5 simultaneous outbound data streams.

5.3 Data Block Reception

The SIV will minimally support data block reception of 1 block per tenth of a second for blocks up to the maximum size of a LAN packet.

The SIV will respond to data block reception processing controls within 1 second of the receipt of the request.

The SIV will support 5 simultaneous inbound data streams.

The SIV will be capable of logging TBD data (?dependent on size of disk which may differ from test lab to test lab).

5.4 Validation

The SIV will minimally support validation of 1 data stream without interference to concurrent data generation and reception logging.

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 6

RELIABILITY, MAINTAINABILITY, AND RELATED REQUIREMENTS

6.1 Reliability

The SIV test tool will be available at the DTF-21 facility and support at least one tester during a scheduled test time.

The SIV software will be available for installation in machines at the tester's development lab for platforms supported under Section 6.3, Portability.

6.2 Maintainability

In order to achieve maximum maintainability, the SIV software will:

a. Be modular in structure according to function.

b. Contain comments that are concise and consistent with code function.

c. Maximize use of DSCC Existing Software including Multiuse Software (MSW).

d. Maximize use of SPC Data Systems Modification Task's programming language of choice (in this case 'C').

e. The developed 'C' program will comply with the 'C' coding standards of the SPC Data Systems Modification Task.

f. The SIV software will be table driven. That is, maximize the use of tables, both, as parameter storage and to control program flow.

6.3 Portability

The SIV software will be written in 'C' and, as a high order language, assures maximum portability. In addition, it is intended that the program set be developed on the Modcomp 9735 Realix B.1 and SUN Workstation platforms during Build 1. Procurement of a SIV workstation to reside at DTF-21 is planned for Build 2. The Build 3 goal is for SIV to support multiple platforms including, at a minimum, those currently supported by the full MSW program set.

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 7

DELIVERY, INSTALLATION, AND ENVIRONMENTAL REQUIREMENTS

This section specifies incremental program set delivery requirements. The purpose of the incremental approach is to facilitate 'pipeline' testing in concurrence with Subsystem Integration (described in the Subsystem Integration Test Plan). To that end, software analysis in the Software Specification Document (SSD-1) will provide a description of those modules required for stepwise development of the increments described herein.

Current plans for build and installation are defined in Telos's Configuration Management Plan. Packaging of deliverables is also defined there. No special site preparation is required.

Training is not part of this requirement package.

The SIV SUN Workstation will be located in the DTF-21 lab. SIV software will be made available through SPMC for use by developing subsystems whereby SIV software can be downloaded to a development machine in other (non-DTF) labs.

7.1 Delivery Phases

The delivery schedule for SIV software is dominated by external rather than technical constraints. The incremental build schedule is primarily driven by the development schedules of those interfacing subsystems able to benefit from SIV within the 1994 and 1995 fiscal years.

The fundamental philosophy of implementation is incremental; that is, a functional skeleton of the system is developed initially with increasing levels of capability developed in distinct phases. Three builds of the SIV are planned. The phases of implementation for Build 1 include:

a. Prototype Development on the Modcomp 9735

   Demonstrate that the SIV has the capability to establish a dedicated communication path with one interfacing test assembly (ITA), read in at least one SIV Interface Definition (RID) File, generate data blocks for 890-131 or DFL-1-1 delivery, receive and decode 890-131 or DFL-1-1 data blocks, generate ASCII dumps and reports, respond to directives to control data generation/reception/validation processes, and provide for automated 'batch' mode entry of test directives. The SIV will be able to provide displays and event notifications at the local terminal for monitoring of interface tests.

b. Port to SUN Workstation and Completion of Build 1

Port the prototype developed in (a) to a secondary (target) platform--Sun Workstation running Solaris Unix O/S, fine tune the SIV functionality listed in (a), and demonstrate support of targeted Interface Agreements to the customer.

c. Increased Functionality for Subsequent Builds

A minimum of two subsequent SIV builds will support increased SIV functionality. Each subsequent build will include identification and prioritization of SIV requirements based on user feedback, build planning, and implementation; and user visibility and participation through demonstrations and training.

SECTION 8

DESIGN AND IMPLEMENTATION CONSTRAINTS

8.1 Target Environment

SIV is a test and simulation tool not intended for use in the Operational Environment. Rather, SIV is intended for use by software developers (e.g., Cognizant Development Engineer) and operations test personnel (e.g., Cognizant Operations Engineer) in the testing of software interfaces prior to transfer.

To support both types of SIV users, SIV software can be used in development labs as well as at the Development Test Facility (DTF-21). Figure 2-2 presented the anticipated configuration of the SIV for both of these environments.

When SIV software is desired for use in a development lab, the user must make available a machine onto which SIV software will be loaded for operation. The machine selected must be of a platform supported by SIV. Note: Current MSW limitations restrict simultaneous execution within the same machine of more than one application built with MSW.

The SIV communicates with the interfacing subsystem via the SPC LAN. The local terminal is connected to the SIV.

8.1.1 Target CPU

The SIV is targeted for multiple platforms. The initial SIV prototype will be developed on a Modcomp 9735 (68K) running Realix O/S. By the end of Build 1 development, SIV will support both the prototype platform as well as the Sun Sparc Workstation running Solaris O/S.

Subsequent SIV builds will support additional platforms as user need is identified. The selection and number of platforms supported will be similar to that supported by MSW.

8.1.2 Target Language

The language of choice for development of the SIV software is the "C" programming language. This should allow for maximum portability as the development is targeted for multiple platforms.

8.1.3 Use of Existing Software

The SIV will utilize Multiuse Software (MSW) that is currently in use for other SPC Upgrade Subsystems. Software expected to be used within SIV includes:

System Initialization
Shared Memory Services
DSN Protocol and LAN Support
DMC and Operator Interface (ODs, Displays, Monitor Data, Configuration Processing (CCN's), Alarm and Event Processing).
Buffer and Queue Management Utilities with Router/Dispatcher
Autotester Services for Batch Mode Testing
System Utilities (Time Conversion, Data Conversion, Disk I/O, etc.)

The SIV will utilize the NOCC Datagen software which provides a flexible data block generation capability. The SIV will convert this software for portability to multiple platforms and use with MSW.

The SIV may utilize other SPC Upgrade software which may be similar in function such as the tracking data recording, command interface simulation, etc.

8.2 Inherited Design

The Command Simulator test tool concept was adapted for generic use in testing other subsystem interfaces.

8.3 Assumptions

None.

SECTION 9

ACCEPTANCE CRITERIA AND QUALIFICATION METHODS

During Development, several methods will be used to verify the integrity and functionality of the SIV software. These methods include:

a. Unit testing each module before adding it to the skeleton in configuration control. Methods specified by the SIV CDE will be followed. The SIV CDE reserves the right to review test results.

b. Integration testing of unit modules. The automatic tester software will be used as part of the integration testing.

c. Software Demonstrations of development increments identified in Section 7. When possible, Integration Tests by the development organization will include interfacing subsystem developers for familiarization purposes.

d. Acceptance will be based on functional demonstration of the SIV in accordance with the Software Acceptance Test Criteria set forth by an independent test conductor and witnessed by Quality Assurance.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A

TRACEABILITY

The Subsystem Interface Verifier (SIV) Test Tool Requirements have been derived from the SIV Statement of Work and subsequent requirements review meetings attended by JPL management, software developers, and the SIV user community. The following lists software requirements as written in this document and provide the priority level defined in Section 1.8. A cross reference to the Functional Requirements of Section 1.9 is also provided.

REQMT ID	SECTION	REQUIREMENT DESCRIPTION	PRIORITY	FUNC REQMT
	4	PROGRAM SET FUNCTIONAL REQUIREMENTS	n/a	
	4.1	SYSTEM INITIALIZATION	n/a	
SIV001	4.1	Use MSW to perform standard initialization functions	1	OT3
	4.2	CONFIGURE FOR TEST	n/a	
SIV002	4.2	Load subsystem specific information from file into the SIV system	1	MC1
SIV003	4.2	Load SIV Interface Definitions (RIDs) into shared memory	1	MC5
SIV004	4.2	Provide OD to initiate SIV configuration process	1	MC5
SIV005	4.2	Make Configuration data available to other SIV processes	1	MC5
	4.3	DMC EMULATION	n/a	
SIV006	4.3	Emulate DMC FAT and CCN transmissions in order to establish a dedicated communication path with the interfacing subsystem	1	MC1
SIV007	4.3	Provide watch dog service via communications with interfacing subsystem	3	MC1
SIV008	4.3	Emulate LMC functions to process ODs, Displays, and ENs from interfacing subsystem	1	MC1, MC2, MC6
SIV009	4.3	Provide for Autotester use of interfacing subsystem ODs and ENs	2	MC7, MC10, MC1
	4.4	Operator Directive Handling	n/a	
SIV010	4.4	Accept and process directives entered from the	1	MC2,

REQMT ID	SECTION	REQUIREMENT DESCRIPTION	PRIORITY	FUNC REQMT
		local terminal		MC5
SIV011	4.4	Accept and process directives received from the autotester	1	MC10
SIV012	4.4	Accept and process directives to control the data generation process (START STRM, END STRM, PAUSE, etc.)	1	MC5
SIV013	4.4	Accept and process directives to control data logging processes	3	MC5
SIV014	4.4	Accept and process directives to control the data validation process (VALIDATION ON/OFF, REPORT CTRLS, etc.)	2	MC5
SIV015	4.4	Accept and process directives to dynamically modify the contents of data blocks generated (CHG VALUE, CHG ACTION, etc.)	2	MC8
SIV016	4.4	Accept and process directives to dynamically modify data validation parameters per item (CHG RANGE, VALIDATE ON/OFF, etc.)	2	MC8
SIV017	4.4	Accept and process directives to manipulate SIV displays	1	MC2
SIV018	4.4	Accept and process interfacing subsystem's directives	1	MC1
SIV019	4.4	Accept and process directives to dynamically change the SIV test configuration	2	MC5
	4.5	DATA BLOCK GENERATION	n/a	
SIV020	4.5	Build and transmit data blocks based on definition data	1	DG1, DG2
SIV021	4.5	Support standard 890-131 and DFL-1-1 protocols	1	NC3, NC4
SIV022	4.5	Respond to data generation control ODs	1	MC5
SIV023	4.5	Respond to dynamic changes in definition data	1	MC8
SIV024	4.5	Provide visibility into data block content	1	MC2, MC12
SIV025	4.5	Maintain data block generation statistics for reports/displays	2	MC9
SIV026	4.5	Generate blocks which correspond to the test subsystem's incoming interface specifications	1	DG1, DG2
	4.6	DATA BLOCK RECEPTION	n/a	
SIV027	4.6	Accept data blocks in standard 890-131 and DFL-1-1 protocols	1	RV1

REQMT ID	SECTION	REQUIREMENT DESCRIPTION	PRIORITY	FUNC REQMT
SIV028	4.6	Log the raw data (in realtime) to disk	2	RV6
SIV029	4.6	Decode to identify block type and distribute for validation accordingly	1	RV2
SIV030	4.6	Respond to data block reception control ODs	1	MC5
SIV031	4.6	Maintain data block reception statistics for reports/displays	3	MC9
	4.7	DATA VALIDATION	n/a	
SIV032	4.7	Identify corresponding interface definition data and lookup validation parameters	2	RV2
SIV033	4.7	Generate ASCII dump of data to disk	1	MC4
SIV034	4.7	Validate the data block according to validation parameters in RID data	2	RV2
SIV035	4.7	Generate ASCII validation report	3	RV4
SIV036	4.7	Respond to data block validation control ODs	1	MC5
SIV037	4.7	Respond to dynamic changes in RID validation parameters	2	MC8
SIV038	4.7	Store decoded data in format necessary for displays	3	MC12
SIV039	4.7	Maintain validation statistics for reports/displays	3	MC9
	4.8	AUTOMATED TESTING	n/a	
SIV040	4.8	Utilize existing automated capabilities provided by MSW	1	OT3, MC10
SIV041	4.8	Support the interfacing subsystem's ODs and ENs	2	MC1, MC6
	5	PERFORMANCE	n/a	
	5.1	SIV CONFIGURATION	n/a	
SIV042	5.1	Configure for an interface test within 1 second of receipt of the configuration directive	3	derived
	5.2	DATA BLOCK GENERATION	n/a	
SIV043	5.2	Be able to generate blocks at a rate of one block per tenth of second	1	derived (DG8)
SIV044	5.2	Respond to data block generation processing controls within 1 second of receipt	1	derived
	5.3	DATA BLOCK RECEPTION		
SIV045	5.3	Be able to receive blocks at a rate of one block	1	derived

REQMT ID	SECTION	REQUIREMENT DESCRIPTION	PRIORITY	FUNC REQMT
		per tenth of second		(RV2)
SIV046	5.3	Respond to data block reception processing controls within 1 second of receipt	1	derived
	6	RELIABILITY, MAINTAINABILITY, & RELATED REQUIREMENTS	n/a	
SIV047	6.1	Make SIV software available for development lab use	1	derived (OT1)
SIV048	6.1	A SIV system including hardware and software will be available at DTF-21.	1	derived (OT1)
	6.2	MAINTAINABILITY	n/a	
SIV049	6.2	Maximize use of table driven methods	1	derived
SIV050	6.2	Maximize use of existing software	1	OT3
	6.3	PORTABILITY	n/a	
SIV051	6.3	Support multiple platforms including, at a minimum, the popular platforms currently supported by MSW	3	OT1

APPENDIX B

GLOSSARY

Data Generation: The process by which SIV sends a stream of simulated data blocks to the Interface Test Assembly.

Data Validation: The process by which SIV accepts and checks the contents of a given data block against its corresponding Interface Definition Data.

Interface Definition Data: Information which defines, to a bit-level, the format and contents of blocks for a given interface agreement for ingest by the SIV.

Interface Test Assembly: Subsystem or assembly with which SIV is communicating in order to test interface flows.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

ACRONYMS

CCN	Configuration Change Notice
CDE	Cognizant Development Engineer
CMC	Complex Monitor and Control Assembly
COE	Cognizant Operations Engineer
CSW	Common Software
DG	Display Graphics
DGC	Display Graphics Control
DGR	Display Graphics Reject
DMC	DSCC Monitor and Control Subsystem
DOY	Day of Year
DR	Directive Response
DSCC	Deep Space Communications Complex
DSN	Deep Space Network
DTF	Development and Test Facility
EN	Event Notification
FAT	Functional Address Translation
FDD	Functional Design Document
FRD	Functional Requirements Document
GCF	Ground Communication Facility
ID	Identifier
ISB	Intersubsystem Block
ISO	International Standards Organization
ITA	Interface Test Assembly
JPL	Jet Propulsion Laboratory
LAN	Local Area Network
LMC	Link Monitor and Control
MD	Monitor Data
MDR	Monitor Data Request
MDTS	Monitor Data Transfer Segment
MFD	MDTS Format Definition
MFDR	Monitor Data Format Definition
MSS	Monitor Segment Specification
MSSR	Monitor Segment Specification Request
MSW	Multi-use Software

M&C	Monitor and Control
OD	Operator Directive
OS	Operating System
O/S	Operating System
OSI	Open Systems Interconnection
RID	SIV Interface Definition
SCOE	System Cognizant Operations Engineer
SHM	Shared Memory
SIV	Subsystem Interface Verifier
SPC	Signal Processing Center
SRD	System Requirements Document
SSE	Subsystem Engineer
S/S	Subsystem
TCT	Time Code Translator